I. J. Communications, Network and System Sciences, 2009, 1, 1-89
Published Online February 2009 in SciRes (http://www.SciRP.org/journal/ijcns/).

On the Scalable Fairness and Efficient Active Queue Management of RED

Hui WANG[1], Xiao-Hui LIN*[1], Kai-Yu ZHOU[2], Nin XIE[1, 4], Hui LI[3]
[1]*Department of Communication Engineering, Shenzhen University, Shenzhen, China*
[2]*China Telecom Beijing Research Institute, Beijing, China*
[3]*Shenzhen Graduate School, Peking University, Shenzhen, China*
[4]*National Mobile Communications Research Laboratory, Southeast University, Nanjing, China*
*Email: *xhlin@szu.edu.cn*
Received October 22, 2008; revised December 2, 2008; accepted December 31, 2008

Abstract

Internet routers generally see packets from a fast flow more often than a slow flow. This suggests that network fairness may be improved without per-flow information. In this paper, we propose a scheme using Most Recently Used List (MRUL)−a list storing statistics of limited active flows that sorted in most recently seen first mode-to improve the fairness of RED. Based on the list, our proposed scheme jointly considers the identification and punish of the fast and unresponsive fast flows, and the protection of slow flows. Its performance improvements are demonstrated with extensive simulations. Different from the previous proposals, the complexity of our proposed scheme is proportional to the size of the MRUL list but not coupled with the queue buffer size or the number of active flows, so it is scalable and suitable for various routers. In addition, another issue we address in this paper is queue management in RED. Specifically, we replace the linear packet dropping function in RED by a judicially designed nonlinear quadratic function, while original RED remains unchanged. We call this new scheme Nonlinear RED, or NLRED. The underlying idea is that, with the proposed nonlinear packet dropping function, packet dropping becomes gentler than RED at light traffic load but more aggressive at heavy load. As a result, at light traffic load, NLRED encourages the router to operate in a range of average queue sizes rather than a fixed one. When the load is heavy and the average queue size approaches the pre-determined maximum threshold (i.e. the queue size may soon get out of control), NLRED allows more aggressive packet dropping to back off from it. Simulations demonstrate that NLRED achieves a higher and more stable throughput than RED and REM. Since NLRED is fully compatible with RED, we can easily upgrade/replace the existing RED implementations by NLRED.

Keywords: Random Early Detection, TCP, Unresponsive Flows, Fairness, Queue Management

1. Introduction

With the increasing popularity of stream media applications, the fairness of networks has attracted much research attention [1−11]. With these research efforts, a number of schemes [4−12] were proposed to improve the fairness in networks with modifications to the queue management schemes implemented in Internet routers.

Known as Active Queue Management (AQM), Random Early Detection (RED) [13] is recommended by IETF for queue management in routers. However, past work shows that unfairness of RED may occur with RED under two conditions. Firstly, when two or more TCP flow with different RTT competing for the bottleneck bandwidth, RED tends to let the flow with shorter RTT use more bandwidth [9]. Secondly, when responsive TCP flows shares a RED router with unresponsive UDP flows, unresponsive UDP flows may has unreasonable high throughput than TCP flows [8].

Although per-flow queue (i.e. Fair Queuing [4,12]) is the most direct solution to the unfair problems, with the large number of flows possibly sharing a link, it is not scalable for an Internet router. Notice that a router sees packets from a fast flow more often than a slow flow; we propose in this paper the Scalable Fair Random Early

Detection (SFRED) to improve the fairness of RED. A Most Recent Used List (MRUL) storing up to *N* most active connections' traffic statistics is maintained by SFRED. Based on the list, SFRED has jointly considered the punishment of fast flows, unresponsive fast flows, and the protection of short life slow flows (e.g. WEB applications). Simulations demonstrate that SFRED has significantly improved the fairness of RED. The complexity of SFRED is proportional to the size of the MRUL but not coupled with the queue buffer size or the number of active flows, so it is scalable and suitable for various routers.

Another issue we address in the paper is efficient queue management in RED. Among various AQM schemes, RED is probably the most extensively studied. RED is shown to effectively tackle both the global synchronization problem and the problem of bias against bursty sources. Due to its popularity, RED (or its variants) has been implemented by many router vendors in their products (e.g. Cisco implemented WRED). On the other hand, there is still a hot on-going debate on the performance of RED. Some researchers claimed that RED appears to provide no clear advantage over drop-tail mechanism. But more researchers acknowledged that RED shows some advantages over drop-tail routers but it is not perfect, mainly due to one or more of the following problems.

- RED performance is highly sensitive to its parameter settings. In RED, at least 4 parameters, namely, maximum threshold (max_{th}), minimum threshold (min_{th}), maximum packet dropping probability (max_p), and weighting factor (ω_q), have to be properly set.
- RED performance is sensitive to the number of competing sources/flows.
- RED performance is sensitive to the packet size.
- With RED, wild queue oscillation is observed when the traffic load changes.

As a result, RED has been extended and enhanced in many different ways. It can be found that a common underlying technique adopted in most studies is to steer a router to operate around a fixed target queue size (which can either be an average queue size or an instantaneous queue size). There are some concerns on the suitability of this approach, since the schemes thus designed are usually more complicated than the original RED. This renders them unsuitable for backbone routers where efficient implementation is of primary concern. In some schemes, additional parameters are also introduced. This adds extra complexity to the task of parameter setting. Unlike the existing RED enhancement schemes, we propose to simply replace the linear packet dropping function in RED by a judiciously designed nonlinear quadratic function. The rest of the original RED remains unchanged. We call this new scheme Nonlinear RED, or NLRED. The underlying idea is that, with the proposed nonlinear packet dropping function, packet dropping is gentler than RED at light traffic load but more aggressive at heavy load. Therefore, at light traffic load NLRED encourages the router to operate in a *range* of average queue sizes rather than a fixed one. When the load is heavy and the average queue size approaches the maximum threshold max_{th}–an indicator that the queue size may soon get out of control, NLRED allows more aggressive packet dropping to quickly back off from it. Simulations demonstrate that NLRED achieves a higher and more stable throughput than RED and REM, an efficient variant of RED. Since NLRED is fully compatible with RED, we can easily upgrade/replace the existing RED implementations by NLRED.

The rest of this paper is organized as following. In Section 2, we introduce the background and the related work of RED, together with the proposed SFRED algorithm. In Section 3, we give the simulation results of SFRED. In Section 4, we present Nonlinear RED. This is followed by extensive simulations in Section 5. Finally, we give the concluding remarks in Section 6.

2. Scalable Fair RED

RED [13] provides high throughput while keeping short queue length (i.e. queuing delay) at the routers. However, [8,9] have shown that RED has fairness problems. When two or more TCP flow with different RTT competing for the bottleneck bandwidth, RED tends to let the flow with shorter RTT use more bandwidth [9]. Similarly, when responsive TCP flows shares a RED router with unresponsive UDP flows, unresponsive UDP flows may has unreasonable high throughput than TCP flows [8].

LRU-RED [4] was developed based on a LRU list to identify high bandwidth flows. This scheme derived from the fact that a router should see a packet form a fast flow more often than a slow flow, so that the number of states to be kept for maintaining the fairness can be bounded. However, we find the design of LRU-RED in [5] is too rough to fully utilize the potential of LRU table. For example, it cannot identify an unresponsive flow so that the second unfairness condition mentioned in Section 1 cannot be solved. Motivated by this, in this paper, we will propose a new scalable fair AQM scheme, the SFRED. SFRED is developed based on a list similar to the LRU but the list (MRUL) keeps more information. SFRED has combined many novel ideas in previous work in its design, such as the punishment of unresponsive flows [8,10] and the protection of slow flows [2].

The fairness of Scalable Fair RED (SFRED) is enforced in three steps, namely, *identifying and limiting fast flows, identifying and punishing unresponsive fast flows, and protecting slow flows*. In this paper, we consider that the SFRED working in "packet mode." That is, all the computations of throughput and bandwidth allocation are in packets. However, the proposed mechanisms can be easily extended to work with all the throughputs and bandwidth allocations computed in bits (i.e. "bit mode").

2.1. MRUL

From the viewpoint of fairness, the flows in networks can be a *fast* or a *slow* flow. A fast flow transmits faster than the fair rate and may interfere with the transmission of other flows; a slow flow utilizes no more than the fair bandwidth.

A fast flow has packets arrives at the router more often than a slow flow. In other words, given the set of fast flows in a router, with packets in the queue sorted with their arrival time, we should able to find at least one packet from each of the fast flow before reaching the head of the queue (searching the queue from the end, i.e. most recently received packet, to the head i.e. lest recently received packet). This suggests that it is possible to build a scalable fair queue management algorithm with limited complexity. Notice that a fair queue management scheme concerns only the aggregated traffic characteristics and the characteristics of the fast flows. The aggregated traffic characteristics are needed for determining the fair rate. The characteristics of fast flow are needed for determining the per flow punishment. Motivated by this, we develop the Scalable Fair RED (SFRED) based on the Most Recent Used List (MRUL).

In SFRED, a router maintains a linked list (MRUL) for up to *N* most recently seen active flows (flows that has packet routed through the router recently). This linked list keeps simple traffic statistics for each of the active flows in list, such as the number of packet received (H_{ri} , i =1,2,…,N) and dropped (H_{di} , i = 1,2,…,N). Besides, SFRED also keeps the number of packet received (H_r), dropped (H_d) for the aggregated traffic, and the total number of activate flows (n).

The MRUL is maintained as follows. Upon receiving a packet, SFRED searches the list for a node matching the address of the arrived packet. If it is not found, SFRED creates a new item for the flow as the list header, and the new item is initialized with H_r =1 and H_d =0. Otherwise, if a node matching the address is located, SFRED increases the number of packet received (H_r) by one and moves this item to the list header. SFRED then processes and checks if the packet should be dropped, and changes the number of packet dropped (H_d) accordingly. When there are already *N* nodes in the list (i.e. the list is full), SFRED deletes the tail node before creates the new one.

Based on the data stored in MRUL, SFRED performs the identification of slow, fast, and unresponsive fast flows, as well as determines the particular punishment to the fast and unresponsive fast flows.

2.2. Identify Fast Flow

To be fair, a queue management scheme should first be able to identify the fast flow. In SFRED, the fair rate T_f is computed as:

$$T_f = \frac{H_r - H_d}{n}$$

With the MRUL, the throughput of an active flow can be shown with the number of packet received (H_{ri}) as

$$T_i = H_{ri}$$

So a flow is a fast flow, if

$$T_i > (1+\alpha)T_f \quad \textbf{(1)}$$

where α is a constant set to 0.1 in this paper.

Once a fast flow is identified, packets from this flow are processed by normal RED with the loss probability increase by T_f / T_i times, as

$$max_{pd} = (T_f max_p) / T_i$$

where max_p is the original maximum dropping probability of RED, and max_{pd} is the maximum dropping probability for the fast flow.

Generally, such increases in loss probability is sufficient for a responsive TCP flow, since the TCP analytical models [14] indicates that TCP throughput is inversely proportional to the packet loss rate. Considering the bursty nature of TCP transmission, this is also necessary as applying a strict bandwidth limitation (e.g. dropping all packets received from a flow except the first five in each 0.5s interval) results in lower than fair rate throughput for a TCP flow.

2.3. Punish Unresponsive Fast Flows

However, simply increasing the dropping probability has been proved to be not effective for the unresponsive flows [8], such as consistent bit rate user datagram protocol flows (CBR-UDP). An unresponsive flow does not dynamically change its throughput with network state (e.g. the packet loss rate). In other words, it does not adopt the similar congestion control mechanisms as TCP. Thus, to maintain the fairness of networks, when there are unresponsive fast flows, the queue management scheme should take a more actively part in punishing them.

In SFRED, the identification of unresponsive fast flows is based on analyzing the drop history of each fast flow that performs similar with the method adopted to identify fast flows in [10]. Notice that an unresponsive fast flow does not changes its transmission rate with the packet loss rate. When it shares a SFRED queue with some responsive flows, comparing with the responsive flows, it has a higher packet loss rate as well as a higher transmission rate. On the contrary, a TCP flow cannot maintain high throughput under a high loss rate. Thus an unresponsive fast flow is identified by comparing the per-flow/average loss rate and the per-flow/average throughput. From sub-section 2.1, the packet loss rate for a flow is

$$L_i = H_{di} / H_{ri}$$

The average packet loss rate is $L = H_d / H_r$.

Thus, when (1) and $L_i > (1+\beta)L$,

where β is a constant set to 0.1 in this paper, a unresponsive fast flow is identified. The identified flow is then applied with a deterministic packet loss

probability of 1. In other words, if a flow has a packet income rate higher than fair bandwidth and at the same time its packet drop rate is higher than the average loss rate, the flow is identified as an unresponsive fast flow and all the subsequent packets from the flow are dropped.

2.4. Protect Slow Flows

The requirement of protecting slow TCP flows, i.e. protecting TCP flows from a packet loss at the slow-start phase, arises from the fact that most short life TCP flows are low slow flows. With limited data to be transmitted, a single short life flow generally cannot reach its full transmission rate before the connection is terminated. However, the performance of short life connection is important for the overall Internet performance.

In SFRED, a new flow seen by the router is protected from experiencing packet loss by modifying the threshold of RED. That is, when a packet is received and the MRUL shows that the flow is with $H_{ri} < \eta$ and zero H_{di}, where η is a constant, the flow is treated as a slow flow. To determine whether the packet should be dropped, SFRED calls RED with both the minimum threshold and maximum thresholds increased by k packets. This equals allocating k packets buffer in the queue to protect slow flow. Because this buffer is still a part of the queue, when congestion becomes severe, slow flows will experience packet loss so that their throughputs are controlled by SFRED. Because the congestion window of a TCP connection reaches a level that able to generate duplicate ACK after send out about 10 packets, the constant η is set to 10.

3. Simulation Validation of SFRED

We evaluate the performance of SFRED by simulating the network in Figure 1 with NS2 [15] simulator. The senders (noted S_i, i=1, …, n) are linked to router G0 with 10Mbps links, with variable propagation delay τ_i ms. A common receiver R is linked to router G1 with a 10Mbps link with delay τ_r ms. From Figure 1, the link between routers G0 and G1 is the bottleneck of the network, with a bandwidth of 1.5Mbps and a delay τ_c ms. Note that all the delays, namely, τ_i, τ_r, and τ_c, are variable. For the RED parameters [13], unless otherwise stated, we use minimum threshold *minth*=5, maximum threshold *maxth*=15, weighting factor w_q = 0.002, and maximum dropping probability max_p=0.1, with a fixed buffer size of 50 packets.

The traffic simulated in the network includes long life FTP, CBR-UDP, and short life Web-like flows. In all the simulations, packet size of 1KB is used for TCP flows (FTP and Web-like). The receiver's advertised window is set to two times of the bandwidth delay products, so that it does not limit the throughput of a flow. For a CBR-UDP flow, it transmits a UDP packet with size X bytes every t_d s, where X is fixed for a flow but may differ between flows. The path from S_i to R (forward path) always carries the data packets, while the reverse path from R to S*i* carries the ACK packets.

The Web-like flows are implemented to acquire enough simulation results with short life TCP connections, and avoid collapsing the simulator with too many resource allocation requests for new connections. A Web-like flow is designed to get a small amount of data (D_w packets) each time. Upon finishing the current transmission, it resets the connection state (so that its transmission restarts with slow start) and transmits another D_w packets.

3.1. Fairness among TCP Flows

Figures 2 and 3 show the bandwidth usage of two FTP flows under RED and SFRED queues, with τ_1=10ms and τ_2=100ms, τ_c=10ms, and τ_r=5ms. Each marker on the curves represents a 10s average of the TCP throughput traced at router G0. With much less margin between the two curves, Figures 2 and 3 show that SFRED achieved fairer bandwidth allocation than RED. Figures 4–6 show the simulation results with 30 Web-like flows and 2 Ftp flows, over 2000s of simulation duration. The simulation parameters are $\tau_1=\tau_2=\ldots=\tau_{30}=\tau_c$=10ms, τ_{31}=10ms, τ_{32} =100ms, τ_r=3ms, and D_w=10 packets. Figure 4 shows the bandwidth usage of the two Ftp flows under the RED queue and Figure 5 shows similar results under the SFRED queue. The dash dotted curve presents the theoretical fair rate-the bottleneck bandwidth over the total number of flows. Again we see SFRED enables much efficient fair bandwidth allocation than RED.

Figure 6 shows the cumulative distribution function of the data transfer delay of the 30 Web-like flows. From this figure, the transfer delay of Web-like flows under SFRED is more stable than RED, in that most of the transmissions finish in 2s. The improvement to transfer delay performance is further presented statistically with the variance and mean in Table 1. It shows that SFRED performs better in the both metrics.

Table 1. Statistic of data transfer delay, 30 web and 2 FTP flows.

	RED	SFRED
mean(s)	2.057	1.825
Var	18.1812	3.977

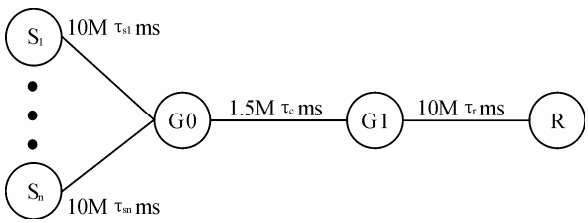

Figure 1. The network simulated.

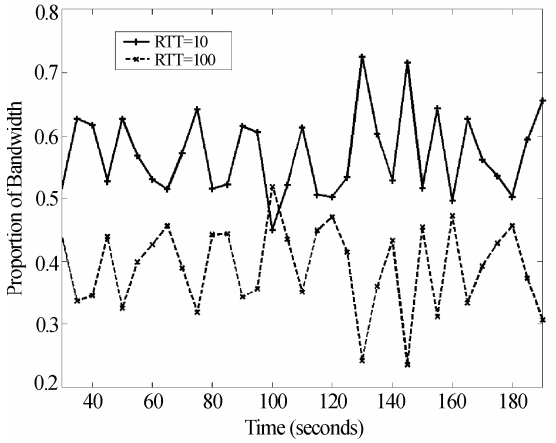

Figure 2. Bandwidth usage of two FTP flows, with τ_1=10ms and τ_2=100ms, under RED queues.

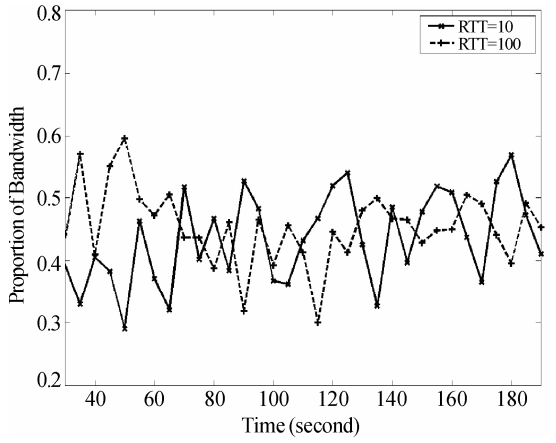

Figure 3. Bandwidth usage of two FTP flows, with τ_1=10ms and τ_2=100ms, under SFRED queues.

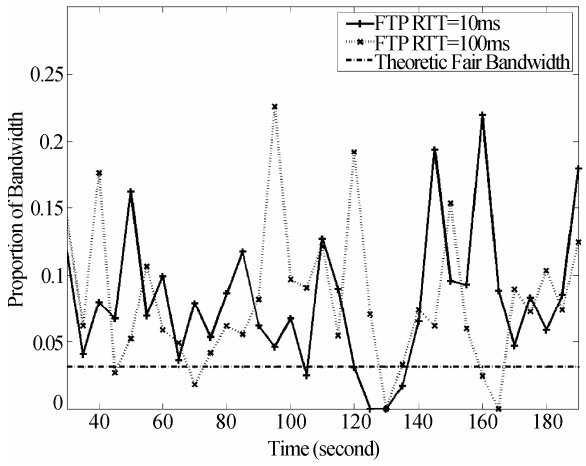

Figure 4. Bandwidth usage of two FTP flows, with τ_1=10ms and τ_2=100ms, under RED queues, with 30 web-like flows.

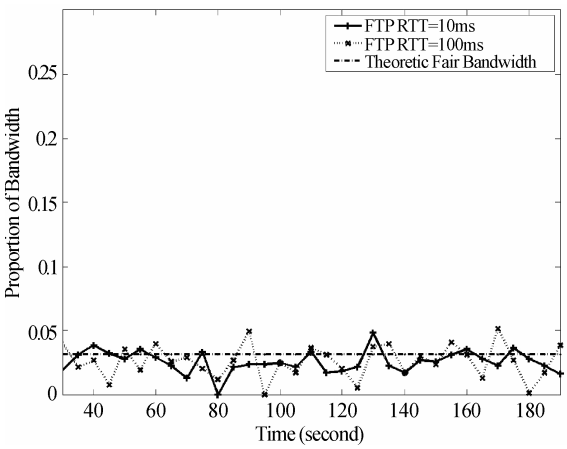

Figure 5. Bandwidth usage of two FTP flows, with τ_1=10ms and τ_2=100ms, under SFRED queues, with 30 web-like flows.

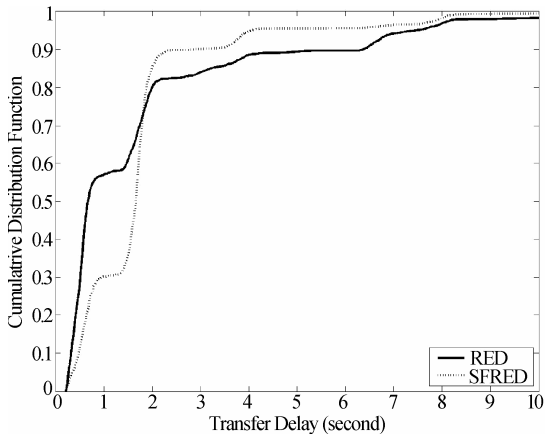

Figure 6. Cumulative distribution function of the transmission delay, 30 web and 2 FTP flows.

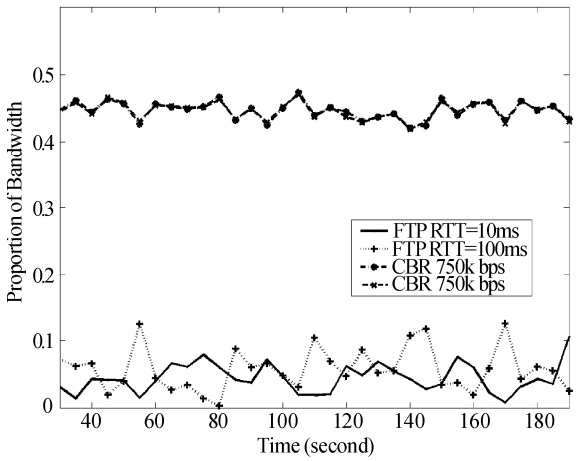

Figure 7. Instantaneous bandwidth usage of two FTP flows and two CBR-UDP flows, under RED queues.

3.2. With CBR-UDP Flows

Figures 7 and 8 show the bandwidth of 4 flows. Two of them are CBR-UDP flows, with τ_1=10ms, X_1=1KB, t_{d1} =10ms, t_2 =100ms, X_2 =500B, and t_{d2} =5ms. The other two flows are Ftp flows, with τ3=10ms and τ_4=100ms. τ_c =10ms and t_r =5ms. From the figures, it is clear that the implementation of SFRED protected TCP flows by punishing the unresponsive UDP flows, although rigorous fairness is still not attained.

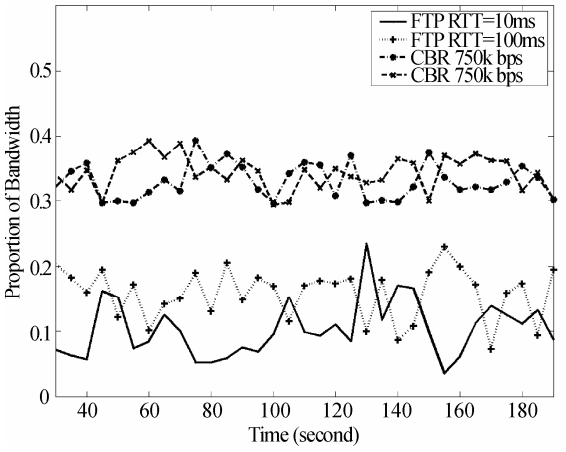

Figure 8. Instantaneous bandwidth usage of two FTP flows and two CBR-UDP flows, under SFRED queues.

4. Nonlinear Random Early Detection

4.1. NLRED Algorithm

RED was mainly designed to overcome the two problems associated with drop-tail routers, namely, global synchronization and bias against bursty sources. Unlike the drop-tail mechanism, RED measures congestion by the average queue size and drops packets randomly before the router queue overflows. When a packet arrives at a router, the average queue size, denoted avg, is updated using the following exponentially weighted moving average (EWMA) function,

$$avg = (1-\omega_q)avg' + \omega_q q$$

where avg' is the calculated average queue size when the last packet arrived, q is the instantaneous queue size, and ω_q is the pre-determined weighting factor with a value between 0 and 1.

As avg varies from a minimum threshold min_{th} to a maximum threshold max_{th}, the packet dropping probability p_d increases linearly from 0 to a maximum packet dropping probability max_p, or

$$p_d = \begin{cases} 0 & avg \le \min_{th} \\ \dfrac{avg - \min_{th}}{\max_{th} - \min_{th}} max_p & \min_{th} \le avg \le \max_{th} \\ 1 & \max_{th} \le avg \end{cases} \quad \textbf{(2)}$$

The throughput performance of RED is not stable. For example, when the traffic load is very light and RED parameters are aggressively set or when the traffic load is very heavy and the parameters are tenderly set, the throughput is low. It has been shown that no single set of parameters for RED could get a stable performance under different traffic loads. We believe such instability is due, at least in part, to the linear packet dropping function adopted by RED, which tends to be too aggressive at light load, and not aggressive enough when the average queue size approaches the maximum threshold max_{th}. We also believe that the performance improvement of some previous work is at least partly due to the employment of nonlinear dropping function, either intentionally or unintentionally. (More reasons to be provided later.) However, we notice that these improvements may not be suitable for core routers, as their corresponding nonlinear dropping functions greatly complicate the basic mechanism of RED. In this paper, we propose to replace the linear packet dropping probability function by a judicially designed quadratic function. The resulting scheme is called non-linear RED or NLRED. The pseudocode of NLRED is summarized in Figure 9.

When avg exceeds the minimum threshold, NLRED uses the nonlinear quadratic function shown in (3) to drop packets, where max_p' represents the maximum packet dropping probability of NLRED. Figure 10 compares the packet dropping functions for RED and NLRED. (The choice of a quadratic function is further explained in the next subsection.)

$$p_d' = \begin{cases} 0 & avg \le min_{th} \\ (\dfrac{avg - min_{th}}{max_{th} - min_{th}})^2 max_p' & min_{th} < avg \le max_{th} \\ 1 & max_{th} < avg \end{cases} \quad \textbf{(3)}$$

Comparing (3) to the dropping function of original RED in (2), if the same value of max_p is used, NLRED will be gentler than RED for all traffic load. This is because the packet dropping probability of NLRED will always be smaller than that of RED. In order to make the two schemes to have a comparable total packet dropping probabilities, we set $max_p' = 1.5 max_p$, such that the areas covered by both dropping functions from min_{th} to max_{th} are the same, or

$$\int_{min_{th}}^{max_{th}} p_d d(avg) = \int_{min_{th}}^{max_{th}} p_d' d(avg)$$

4.2. Why Use a Quadratic Function?

Given that N TCP flows equally share a link with

NLRED

for each packet arrival:
  calculate the average queue size avg
    if $avg \le min_{th}$
    no packet drop
  else if $min_{th} \le avg \le max_{th}$
    calculate the packet drop probability using (2)
    drop the packet with the calculated probability
  else
    drop the packet

Figure 9. Pseudocode of NLRED.

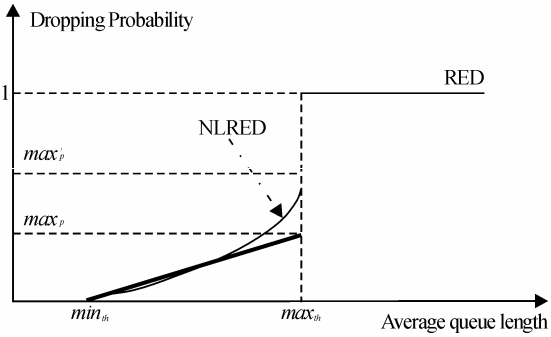

Figure 10. Dropping functions for NLRED and RED.

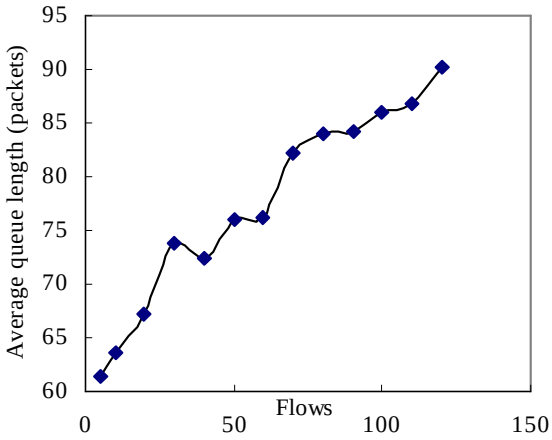

Figure 11. Average queue size vs. number of flows, with drop-tail router.

bandwidth L, and experience a random packet loss/drop probability p. It was shown that p and N has the following relationship.

$$p < \left(\frac{N}{L} \frac{MSS * a}{RTT} \right)^2$$

where α is a constant. This equation indicates that to effectively manage the flows (so as to fully utilize the available network bandwidth) the packet dropping probability should vary quadratically with the number of flows. However, finding the number of active flows N needs 1) per flow information, 2) extra storage space for storing extra state information, and 3) extra router processing overhead. Besides, the resulting flow number is nothing more than an estimation [13,16].

In (3), we have proposed to vary the packet dropping probability based on a quadratic function of average queue size. In [17], it is shown that the average queue size at a router is roughly directly proportional to the number of active TCP flows passing through it. This is further verified by the simulations results shown in Figure 11. The average queue size versus the number of flows is obtained by simulating the network in Figure 12 with drop-tail router mechanisms. (Other simulations using RED with different traffic load also show similar results.)

In fact, choosing a quadratic function is also intuitively

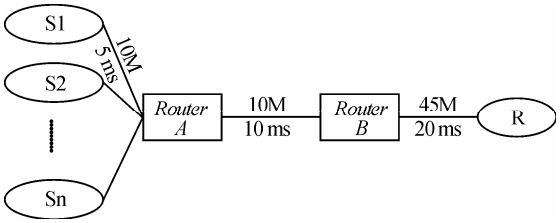

Figure 12. The network simulated.

appealing. From Figure 10, when the average queue size is slightly larger than min_{th}, the packet dropping probability is smaller than the corresponding RED. As such, the average queue size will not be forced to work around min_{th} as strongly as that in RED. Or, one can interpret this as follows. Under current traffic load, the signal for congestion is not strong enough to justify any severe measures to cut back queue size; so a gentler than RED packet dropping probability is desirable. While doing this, we naturally encourage the routers to operate over a range of queue sizes closer to min_{th} (instead of at a fixed target queue size). When avg approaches max_{th}, the congestion becomes more pronounced. The routers can thus take decisive actions to drop packets at a rate higher than RED. When avg is bigger than max_{th}, the routers drop any packets received. Although GRED shows superior performance than RED with an additional linear dropping function when avg is between max_{th} and $2max_{th}$, the design of NLRED does not adopt similar approach. Besides simplifying the algorithm, determined drop is more reasonable for NLRED than another slow changed dropping function (such as used in GRED), because higher than RED dropping probability has already been proven to be too gentle.

5. Simulation of Nonlinear RED

NLRED is implemented using ns-2 simulator [18]. We conduct the simulations based on the network in Figure 12, which consists of N senders and one sink, connected together via two routers A and B. The link between the two routers is the bottleneck. By varying N, we produce different levels of traffic load and thus different levels of congestion on the bottleneck link. The active queue management schemes under investigation are implemented at router A, which has a queue buffer size of 120 packets. Unless otherwise stated, we assume that all packets generated by the senders are 1000 bytes long. Extensive simulations based on this network using different TCP implementations (Tahoe, Reno, and New Reno), RTTs, and AQM schemes (with different parameter sets), are conducted, whereas only a representative subset of the results based on TCP Reno is reported below. Besides, we choose to compare NLRED with GRED [19] instead of RED, due to the superior performance of GRED over RED. We also compare NLRED with REM [18] as it is a representative scheme that steers a router to operate

around a fixed target queue size with excellent reported performance.

***Experiment* 1**

Figures 13 to 16 show the results of a set of simulations with the number of long-lived TCP flows increasing from 5 to 120 and max_p varying from 0.02 to 0.5. The receiver's advertised window of each connection is set to be bigger than the bandwidth delay product. Each point of the simulation results is obtained from a single 200 seconds simulation while the statistics are collected in the second half of the simulation time (i.e. the second 100-second interval).

As explained earlier, in order to compare GRED and NLRED, the maximum packet dropping probability of NLRED is set as $max'_p = 1.5max_p$. As such, the simulation results/curves obtained using NLRED will be labelled by its equivalent max_p instead of max'_p. As an example, the line labelled with $max_p = 0.1$ in Figure 16 means the actual maximum packet dropping probability is $max'_p = 0.15$. Both GRED and NLRED use the same set of parameters, $\omega_q = 0.002$, $min_{th} = 10$, and $max_{th} = 30$.

Figures 13 and 14 show the bottleneck link throughput against the number of flows. Each curve in the figures represents the simulation results with a given max_p. Comparing the two figures, we can see that NLRED is less sensitive to the choice of max_p under different traffic loads (i.e. number of flows). Although the throughput of NLRED still changes with the load, for some max_p selections (e.g. $max_p = 0.05$ to 0.1, or max'_p =0.075 to 0.15), NLRED is very successful in maintaining a high throughput regardless of the loading. This is mainly due to NLRED's nonlinear quadratic packet dropping function, which allows more packet bursts to pass when the average queue size is small, and drops more packets when the average queue size becomes large.

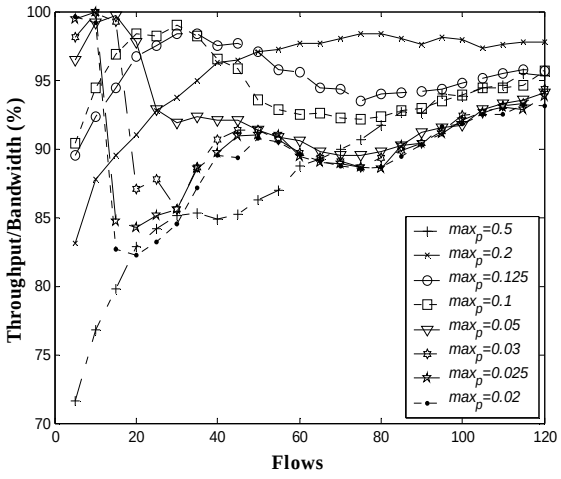

Figure 13. Throughput vs. number of flows using GRED.

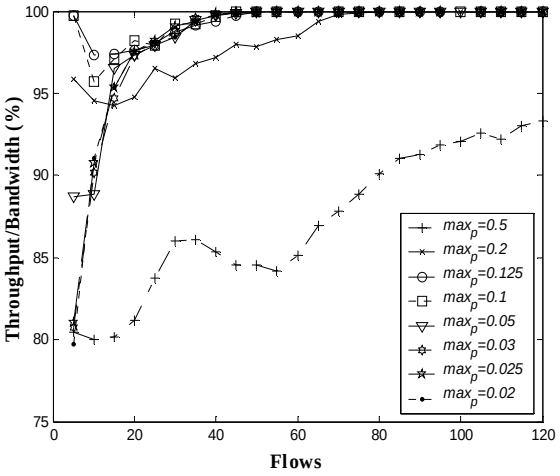

Figure 14. Throughput vs. number of flows using NLRED.

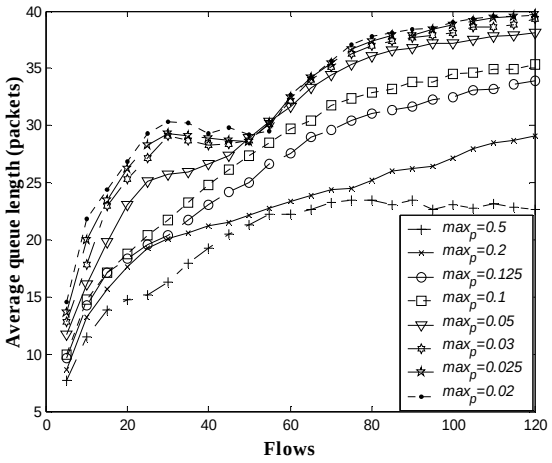

Figure 15. Average queue size vs. number of flows using GRED.

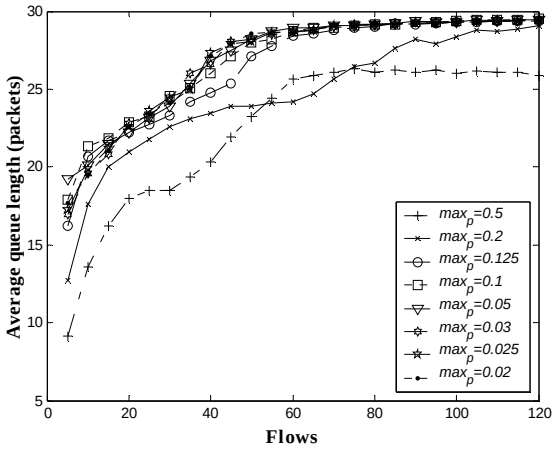

Figure 16. Average queue size vs. number of flows using NLRED.

Figures 15 and 16 show the change of the average queue size with the number of flows. Unlike GRED, we can see that NLRED allows the average queue size to grow at a faster rate when the number of flows is small. As the number of flows increases, NLRED tends to control the average queue size better (i.e. the queue size converges to a stable value faster) than GRED.

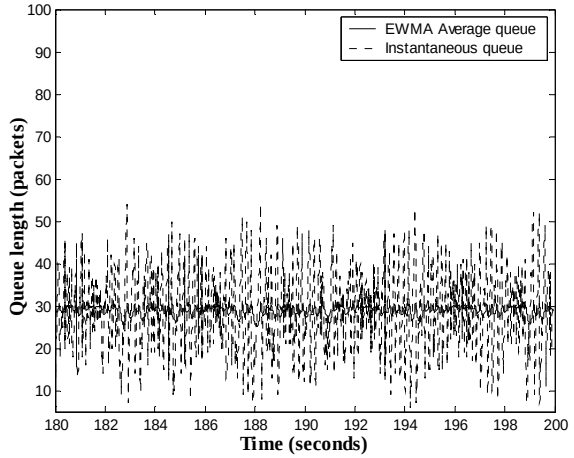

Figure 17. Change in queue occupancy when NLRED is used with *N*=100 flows.

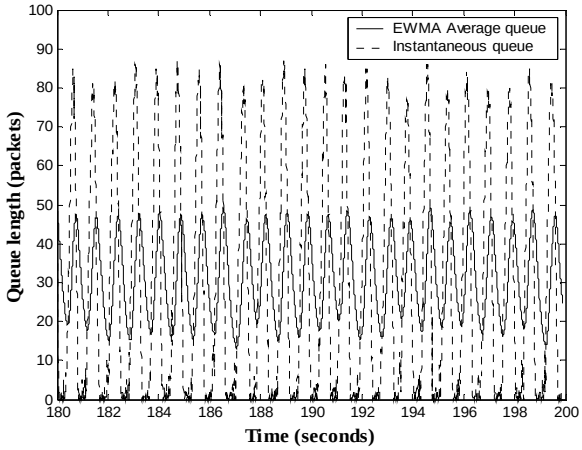

Figure 18. Change in queue occupancy when GRED is used with $N = 100$ flows, $max_p = 0.02$, $\omega_q = 0.002$, $min_{th} = 10$, $max_{th} = 30$.

To have a closer examination on the ability to control queue size, we show in Figures 17 and 18 the instantaneous and average queue sizes against time, with the number of flows $N = 100$ and $max_p = 0.02$. We can see that the oscillations in both instantaneous and average queue sizes are much more noticeable when GRED is used. With NLRED, the oscillations are effectively suppressed, again due to its nonlinear packet dropping function.

***Experiment* 2**

We compare the performance of GRED, REM [20] and NLRED under different traffic loads. We set max_p of all the three AQM schemes to 0.1, $\omega_q = 0.002$, $min_{th} = 10$, and $max_{th} = 30$. The default parameters of REM in ns-2 are used, they are $\gamma = 0.001$, $\alpha = 0.1$, $\phi = 1.001$, and $b = 20$.

From Figure 19, we can see that NLRED has the highest overall throughput, whereas GRED is the lowest.

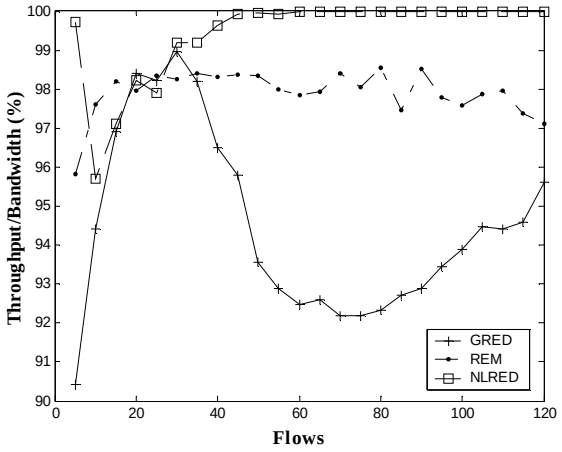

Figure 19. Throughput vs. flow number: GRED, REM and NLRED.

It is interesting to see a short concave phase when the traffic is changed from 10 flows to 40 flows. It is shown that the performance of NLRED is not very stable during this range, partly because of the sharp non-contiguous increase of dropping probability from max_p to 1 when avq grows over max_{th}. However, as soon as the number of flows is larger than 40, the throughput for NLRED quickly converges to the link bandwidth. Besides, during the concave range, the throughput of NLRED is still always higher than GRED. Figure 20 shows the corresponding average queue size of using GRED, REM, and NLRED. By steering the queue around a target length, REM suffers the low throughput when traffic load is extremely light (less than 5 flows) and extremely high. When $N > 60$, the throughput of REM is unstable and drops as *N* increases.

Since Misra, *et al.* [14] indicated that packet size affects the performance of AQM schemes, in this experiment (again based on Figure 12), we test and compare the packet size sensitivity of GRED, REM, and NLRED. REM is configured to work in byte mode because packet mode shows extremely poor performance. (We believe the error

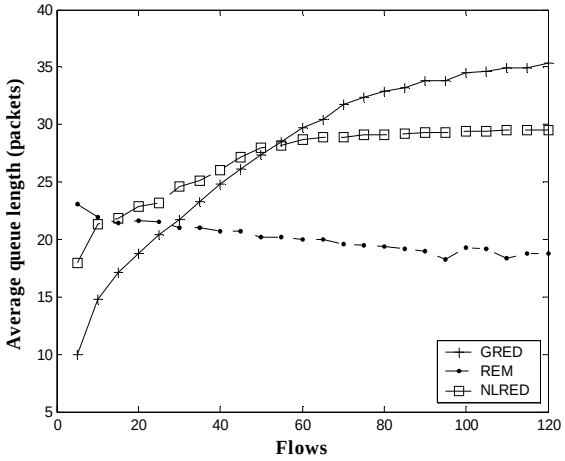

Figure 20. Average queue size vs. flow number: GRED, REM and NLRED.

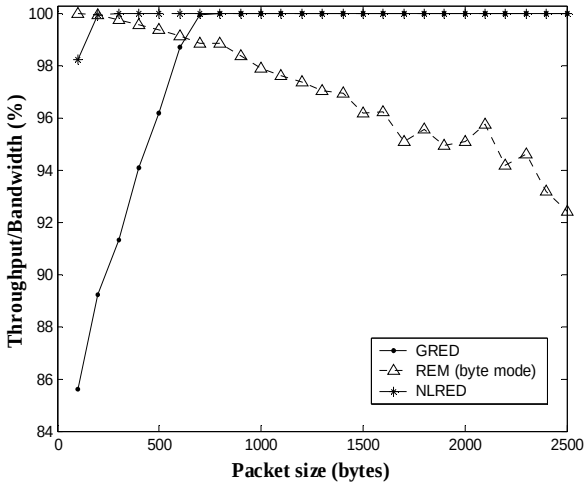

Figure 21. Throughput vs. packet size.

in arrival rate estimation is the reason for such poor performance with packet mode REM.) However, the queue length of byte mode REM cannot be directly compared with the results of other schemes, because it uses bytes as unit whereas others use packets. To solve the problem, we normalize the queue length of byte mode REM to use packets as unit. The conversion assumes all the packets are with the same size as the referenced packet size. We simulate 50 long-lived FTPs. For each AQM algorithm, we conduct a set of simulations with the packet size ranging from 100 bytes to 2500 bytes. Figure 21 shows the throughput against packet size. We can see that NLRED is least sensitive to the packet size and therefore is better than both GRED and REM.

6. Conclusions

We have proposed a mechanism improving the fairness of Internet routers, which called SFRED. The mechanism was developed with a MRUL in which states of up to *N* most recently used flows are stored. SFRED then identifies and punishes the fast and unresponsive fast flows. To improve short TCP transaction performance, SFRED also protects slow flows by allocating a small amount of buffer. Simulations show that the SFRED proposed has significantly improved the fairness of RED, with only limited resource usage. Different from the previous proposals the complexity of SFRED is proportional to the size of the list but not coupled with the queue buffer size or the number of active flows, so it is scalable and suitable for various routers. Moreover, in this paper, we also proposed a new active queue management scheme called Nonlinear RED (NLRED). NLRED is the same as the original RED except that the linear packet dropping probability function is replaced by a nonlinear quadratic function. While inheriting the simplicity of RED, NLRED was shown to outperform RED as well as REM and some of its variants. In particular, NLRED is less sensitive to parameter settings, has a more predictable average queue size, and can achieve a higher throughput. We credit the above performance gain to the idea of encouraging the router to operate over a range of queue sizes according to traffic load instead of at a fixed one. This is realized in NLRED by using a gentle packet dropping probability at the onset of the congestion, and a much more aggressive dropping probability when the congestion becomes more pronounced.

7. Acknowledgement

This work is jointly supported by National Science Foundation of China under the project number 60773203, 60602066 and 60872010, and grant from Guangdong Natural Science Foundation under project number 5010494. The work has also got support from Foundation of Shenzhen City under project number QK200601, open research fund of National Mobile Communications Research Laboratory, Southeast University under project number W200815, and National High Technology Research and Development Program of China under project number 2007AA01Z218.

8. References

[1] F. Ren, C. Lin, and X. Huang, "TCC: A two-category classifier for AQM routers supporting TCP flows," IEEE Communications Letters, Vol. 9, pp. 471−473, 2005.

[2] R. T. Morris, "Scalable TCP congestion control," Ph. D thesis, Harvard University, 1999.

[3] C. V. Hollot, V. Misra, D. Towsley, *et al*., "Analysis and design of controllers for AQM routers supporting TCP flows," IEEE Transactions on Automatic Control, Vol. 47, pp. 945−959, 2002.

[4] M. Nabeshima and K. Yata, "Performance improvement of active queue management with per-flow scheduling," IEEE Proceedings-Communications, Vol. 152, pp. 797−803, 2005.

[5] Smitha and A. L. N. Reddy, "LRU-RED: An active queue management scheme to contain high bandwidth flows at congested routers," in Proceedings of IEEE Global Telecommunications Conference 2001 (GLOBECOM '01), San Antonio, TX, USA, Vol. 1, pp. 2311−2315, November 25−29, 2001.

[6] C. Brandauer, G. Iannaccone, C. Diot, *et al*., "Comparison of tail drop and active queue management performance for bulk-data and web-like Internet traffic," in Proceedings of Sixth IEEE Symposium on Computers and Communications 2001, (ISCC '01), Hammamet, Tunisia, pp. 122−129, July 3−5, 2001.

[7] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED," in Proceedings of Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '99), New York, USA, Vol. 3, pp. 1346−1355, March 21−25, 1999.

[8] F. M. Anjum and L. Tassiulas, "Fair bandwidth sharing among adaptive and non-adaptive flows in the Internet," in Proceedings of Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM

’99), New York, NY, USA, Vol. 3, pp. 1412−1420, March 21−25, 1999.

[9] D. Lin and R. Morris, “Dynamics of random early detection,” in Proceedings of ACM SIGCOMM ’97, Cannes, France, pp. 127−137, September 14−18, 1997.

[10] R. Mahajan, S. Floyd, and D. Wetherall, “Controlling high-bandwidth flows at the congested router,” in Proceedings of Ninth International Conference on Network Protocols, pp. 192−201, 2001.

[11] M. Christiansen, K. Jeffay, D. Ott, *et al*., “Tuning RED for web traffic,” in Proceedings of ACM SIGCOMM 2000, Stockholm, Sweden, pp. 139−150, August 28−September 1, 2000.

[12] B. Suter, T. V. Lakshman, D. Stiliadis, *et al*., “Design considerations for supporting TCP with per-flow queueing,” in Proceedings of Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM ’98), San Francisco, CA, USA, pp. 299−306, March 31−April 2, 1998.

[13] L. Zhang, S. Shenker, and D. D. Clark, “Observations on the dynamics of a congestion control algorithm: the effects of two-way traffic,” in Proceedings of ACM SIGCOMM ’91, the Conference on Communications Architecture & Protocols, Vol. 1, pp. 133−147, Zurich, Switzerland, September 03−06, 1991.

[14] S. Floyd, J. Mahdavi, M. Mathis, *et al*., “An extension to the selective acknowledgement (SACK) option for TCP,” in RFC 2883, 2000.

[15] K. Xu and N. Ansari, “Stability and fairness of rate estimation based AIAD congestion control in TCP,” IEEE Communications Letters, Vol. 9, pp. 378−380, 2005.

[16] T. J. Ott, T. V. Lakshman, and L. H. Wong, “SRED: Stabilized RED,” in Proceedings of Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, (INFOCOM’99), New York, NY, USA, Vol. 3, pp. 1346−1355, March 21−25, 1999.

[17] B. Braden, D. Clark, J. Crowcroft, *et al*., “Recommendations on queue management and congestion avoidance in the Internet,” in RFC2309, April 1998.

[18] V. Misra, W. B. Gong, and D. Towsley, “Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED,” ACM SIGCOMM Computer Communication Review, Vol. 30, pp. 151−160, 2000.

[19] B. Zheng and M. Atiquzzaman, “DSRED: An active queue management scheme for next generation networks,” in Proceedings of 25th Annual IEEE Conference on Local Computer Networks (LCN’00), Tampa, FL, USA, Vol. 1, pp. 242−251, November 8−10, 2000.

[20] S. Athuraliya, S. H. Low, V. H. Li, *et al*., “REM: Active queue management,” IEEE Network, Vol. 15, pp. 48−53, 2001.

